Johann Sampler Bach: Interactive Sampling of Harmonization Computational Models.

Saul Ivan Rivas Vega
Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas
Universidad Nacional Autónoma de México
Ciudad Universitaria, CDMX México
saul.ivan.rivas.vega@comunidad.unam.mx

Abstract

The development and incorporation of computational systems in creative domains can be extended and adapted to embody distinct perspectives originated from taxonomies in human-computer creative interactions. In this work we analyze a system capable of harmonize a melody in style of Bach and by report its limitations and results in measuring various criteria we opted to extend its capabilities by incorporating it as a virtual colleague in the task of music composition. This was done by developing a plug-in for the music composition software MuseScore.

Introduction

In modern creative systems such as COCONET (Huang et al. 2019a), usually is used some kind of machine learning architecture and while these techniques are quite effective the non-technical aspects of it are often neglected. It may not be intentional or completely undesirable for the authors to not do such evaluations, and by any means is implied that would always have a poor performance. In contrast I believe these kind of systems can undergo the commonly used creative evaluations to have a better understanding of its strengths and weakness to be further improved. The results then can be of aid to plan and develop extensions which allow the system to fulfill a goal in computational creativity, being it of incorporating the systems in our society. But how to allow the incorporation in society if the development stops where the only ones able to use them are technical users which can understand the work of the authors. To inspire researchers and interested programmers in develop more welcoming interfaces and at various extents is the intention of this work. Here we aim to extend COCONET capabilities to be able to interact with a user using the music composition software MuseScore, without the need to understand the inner workings of computational storage and analysis of music. This extension then can be seen in a perspective of a colleague or creative support tool depending of the actual purpose of the human composer. In the following sections we review the related work, the evaluation results and limitations finalizing with the developing stages and conclusions of the overall process.

Related Work

Melody Harmonization in music composition has been widely addressed by computational creativity research, most commonly the 4-part harmonization of the baroque era, the one Johann Sebastian Bach was known for. They are diverse in the methodologies used and in the way an user can interact with them. While having an extensive and detailed list of all the previous works is not intended I proceed to mention some of the most recognized works in the proceedings of the International Conference in Computational Creativity (ICCC) and in other popular venues as well. In the proceedings of the ICCC to the best of my knowledge there are 3 publications which integrate this task.

In (Pachet and Roy 2014) Unary Markov Constraints are used, although the main examples used in their publication come from a Jazz style harmonization it is concluded that it can be extended to any type of harmonization given the adequate corpora in MIDI format to be trained on, for example one in the style of Bach. The proposed model in (Meade et al. 2019) uses a Recurrent Neural Network and a dataset consisting of compositions on various styles in MIDI format for the training. They include some easy to understand conditioning controls which allow to specify the length, velocity and density of notes present in the outputs. Lastly in (Bazin and Hadjeres 2019) the main contribution is an interface rather than an actual generative model, but included in this overview as it is stated that has the property of being model-agnostic, meaning that integrated with the proper model it can act as a intuitive harmonization tool powered by ”inpainting” models, which are capable of fill the score accordingly to the desired style given some notes and chords already written.

An early project on music harmonization is EMI (Cope 1992) uses a method defined by Cope as recombinancy which segments Bach chorales, analyze its musical function and rearranges them in a coherent way, being able to produce a full piece. Choral (Ebcioglu 1990) an expert system with over 350 rules is built on a logical programming language running iterations mostly consisting of chord creation an filling. Bach in a Box (McIntyre 1994) addresses the baroque harmonization as a space that can be explored by genetic algorithms given a user-defined initial melody.

Early neural-network approaches include HARMONET (Hild, Feulner, and Menzel 1992) which trains neural networks in order to extract and learn musical features while an algorithm has coded the structural rules that otherwise could be difficult to infer, this system is able to harmonize a initial melody. In (Allan and Williams 2005) a Hidden Markov Model is used, in which the visible states are the notes and the hidden ones are chords, it is capable of generate novel harmonizations. A Deep Long Short-Term Memory Generative Model is used in (Liang et al. 2017) in which there is a special focus on the sampling method which allows not only to harmonize a given input but to create full compositions.

A steerable model is the one present in DeepBach (Hadjeres, Pachet, and Nielsen 2017) where using Dependency networks (a graphical modeling of probabilistic relationships) can be sampled using distinct constraints such as notes, chords, tempo and cadence, which allowed them to being able to develop it also as an add-on to musescore, a score writing software. In the Bach Doodle (Huang et al. 2019b) a scalable web interface was developed to allow users to harmonize a given melody, which was using COCONET (Huang et al. 2019a) as the internal model. COCONET uses a Deep Convolutional Neural Network with Orderless NADE cells, which coupled with blocked-gibbs sampling can fill, continue and create novel compositions.

From the reviewed works we can observe that their capabilities make them subject of a classification such as colleague, where all 12 systems have such interactivity for music creation when having an initial input from the user to work with. As an autonomous agent we have a more reduced list with 6 of them (Pachet and Roy 2014; Cope 1992; Ebcioglu 1990; Allan and Williams 2005; Liang et al. 2017; Huang et al. 2019a) which are the ones capable of complete a coherent full piece from sampling their models or to continue the iterations of their inner processes. Where the role of a coach could be one from which one can infer or learn of the systems' compositional decisions we can think of the 2 following systems (Hadjeres, Pachet, and Nielsen 2017; Meade et al. 2019) which have a more intuitive, traceable and manageable way to determine the conditions to generate the compositions and learn from such outputs.

System Description

COCONET (Huang et al. 2019a) is the model behind the harmonizing Doodle (Huang et al. 2019b), a description according of architectural components proposed by Dan Ventura (Ventura 2017) is used.

Domain: The system works in the domain of music, it provides 4-part harmonization to a given melody or creates its own composition if no user constraint is given.

Knowledge Base: Training and test data are derived from chorale harmonizations by Johann Sebastian Bach. There are 382 chorales with a train/test/valid split (229/77/76) as used in (Allan and Williams 2005).

Genotype: The knowledge base is encoded using a chord vocabulary, but for the actual use in COCONET it was decoded and encoded in a MIDI format and represented as a piano roll as in Figure 1. In the piano roll representation the MIDI values are set in a multidimensional space, with dimension T being the Time, I the instrument, and P the pitch. The smallest note duration is a 16th of a beat, there are 4 instruments and 52 pitches in the MIDI range of (36-88). A value of 1 in a given instrument i, time t and pitch p indicates that the instrument i is playing the pitch p at time t, a 0 is otherwise used for silence.

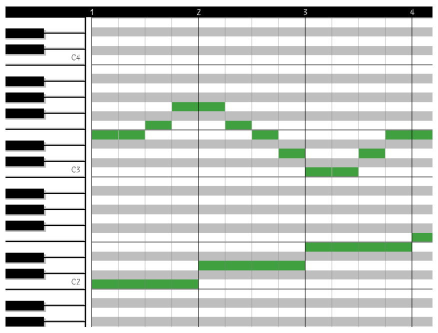


Figure 1: Piano Roll Representation.

Aesthetic and Genotype evaluation: The aesthetic is derived from the dataset, and it is representative of harmonizing as similar to Bach as possible, it is implicit in how the Genotype is evaluated. To evaluate a genotype the Maximum log-likelihood is used, that is evaluate how likely the generated genotype could come from the dataset. In a more detailed way what is modeled in an harmonization is the joint probability of a certain composition given some constraints and during conceptualization the parameters for such probability distributions are estimated as in (Uria et al. 2016).

Conceptualization: As mentioned before, the goal is to estimate the parameters of Categorical Probability Distributions for the dimensions in the piano roll representation. This is accomplished using a Convolutional Neural Network where for each layer the parameters are estimated within the convolutional range of measures in the training dataset transcriptions. Orderless Neural Autoregressive Distribution Estimators (Uria, Murray, and Larochelle 2014) are used as Neural Cells in the Convolutional Network. Some properties derived of using this architecture are that with the CNN, features such as the counterpoint composing technique, can be learned given that it is mainly a local structure rather than a long-term one in the musical transcription. Another property is that with Orderless NADEs not only is possible to estimate parameters for sequential distributions and constraints but also more interesting sampling and environments as well, such as complete a given score, or harmonize an input melody. This is to resemble the compositional task as humans actually do it, not sequentially but by completing partial segments ending up with a full piece.

Generation: Having estimated the parameters for the probability distributions in the conceptualization step now we can sample those distributions in order to get a piano roll with the most probable composition that resembles the ones in the dataset. For such purpose a blocked Gibbs sampling method (Liu 1994) is used. With Independent Blocked Gibbs, as in (Yao et al. 2014), a no necessarily ordered subset of notes, possibly empty, can be used as a query argument to sample the distributions and populate the score. This process is an iterative one, masking some of the already populated notes for a re-sampling, this is repeated several times decreasing the size of the masked subset, in the case of COCONET it is repeated IT times, that is the number of instruments multiplied by the Time slots. This is done due to the quality loss in the sampling for big sized ones, ensuring that the poor estimated samples can be re-sampled yielding a better quality overall in the likelihood evaluation. For comparison a traditional ancestor sampling on NADE is also used for the qualitative evaluation.

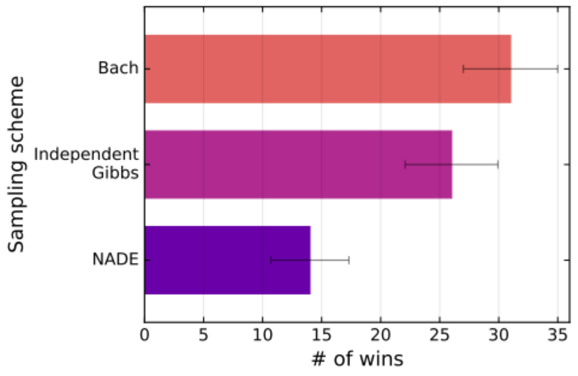


Figure 2: Human evaluations from MTurk on how many times a sampling procedure or Bach is perceived as more Bach-like. Error bars show the standard deviation of a binomial distribution fitted to each's binary win/loss counts.

Phenotype representation and evaluation: To get the phenotype a MIDI interpreter such as the MuseScore Synthesizer or any playback Synth capable of reading MIDI files can be used. The phenotype is the sound of the actual harmonization. For COCONET a listening test was carried on Amazon's Mechanical Turk (MTurk). Generating four unconditioned samples of eight-measure lengths from empty piano rolls. To have an absolute reference for the quality of samples, the first eight measures of four random Bach chorale pieces from the validation set were included. Each fragment lasts thirty-four seconds after synthesis. For each MTurk hit, participants were asked to rate on a Likert scale which of the two random samples they perceive as more Bach-like. A total of 96 ratings were collected, with each source involved in 64 (=96*2/3) pairwise comparisons. Figure 2 shows the number of times each source was perceived as closer to Bach's style.

Creative Limitations

The system has crucial properties for creative systems according to the formal framework of (Wiggins 2006) and from those we can relate the creativity features proposed by (Boden 1991). Such properties are the rules inferred from the dataset in the conceptualization step which define the conceptual space C and the sampling method in the generation step can be treated as T in the framework of Wiggins. The system traversing the static conceptual space C can be related to the Exploratory Creativity in Boden's work. Now this could be accepted in those frameworks but it presents weak points from other perspectives.

Take for instance the argument that it is not creative due merely mimicking Bach, such statement is hard to bypass due to Bach likeliness is explicitly stated as a quality measure for the system to learn without being able to deviate from it at the time of generation.

Other consideration is that although the system can be perceived as one step in the creativity understanding with capabilities such as being a colleague as stated in (Lubart 2005) due to the capabilities of contributing new ideas in a dialog with humans, in this case a musical dialog, like the harmonization of initial musical sequences, and thus comply with 2 of 3 of the conditions proposed by (Pérez 2018). In his work it is claimed that besides of being able to generate novel, coherent and interesting (or useful) products, a creative agent must be able to:

- Employ a knowledge base to build its outcomes.
- Interpret its own outputs in order to generate novel knowledge that is useful to produce more original pieces.
- Evaluate its own products; such an evaluation must influence the way the generation process works.

The system doesn't fulfill the evaluation condition, that is the evaluation doesn't influence generation once the model is trained. Ending with no creative attribution with such conditions.

A third thought on the system comes from an enactive perspective of creativity, in (Davis et al. 2015) it is stated that within enactivism the system's knowledge of the world comes from its own set of experiences, which are strictly constrained to the input of the user, including demonstrations, interactions, and feedback. But such real-time improvisational capabilities are not implemented in the system and in a bigger perspective this would be a great lack on social communication required as stated by (Hertzmann 2018), making it from this approach harder to attribute creativity if any at all.

Overall the system have gaps on its possible attribution of creativity within certain communities that could be addressed in future works expanding its collaboration features. Such expansions could be the ones of an improved interactive interface allowing a modifiable generation process influenced by its own experiences and feedback from the environment.

Evaluation

In this section we will cover the evaluation of COCONET with creativity oriented metrics. Products and Process are evaluated.

Evaluation of the generated products

Measures of novelty and value described in (Pease, Winterstein, and Colton 2001) are used. Novelty and Value are stated properties of the products to be considered creative, based on (Boden 1991).

Novelty Within Alison framework we will use the novelty measure relative to its complexity due to the fact that although not infinite the conceptual space of possible compositions in the piano roll format of length N is quite large. We use the compositions of the 4 voices on each step of the length set to 32 slots each for a 1/16th note. For each step and each instrument can be active any combinations of the note range that is 46. Taking in consideration that a note is active in a given instrument, in a given instrument is denoted by a 1 and 0 otherwise, We can see that the equation to get how many possible outcomes is the following:

$$\begin{aligned} complexity_{i1} = |CS_P| = \\ 2^{N\times Instruments\times Notes} &= 2^{32\times 4\times 46} \\ &= 2^{5888} \end{aligned} \quad (1)$$

Next we evaluate the novelty relative to its complexity. For that 50 samples of length 32 were generated. The possible combinations of notes of the 4 instruments on a given step are our elements k which are used to generate an element $x \in R$. Now we calculate $complexity_{ii2}$, for which first we calculate the frequency $f(k)$ for each $k \in bag_R$ just by reporting the number of occurrences of k in bag_R.
Then the average frequecny(bag_R) is calculated with:

$$af(bag_R) = \frac{|bag_R|}{|set_R|} \quad (2)$$

Now we calculate the relative frequency for each k following:

$$rf(k) = \frac{f(k)}{af(bag_R)} \quad (3)$$

With these results now we can finally calculate $complexity_{ii2}$ for each $x \in R$ by summing all the relative frequencies of all k which are part of x as:

$$complexity_{ii2}(x) = \frac{1}{\sum rf(x)H_x} \quad (4)$$

All the elements share the same $complexity_{i1}$ and we could set an arbitrary value to see if is big enough like $\alpha = 2^{672}$ which is approximately the number of operations a computer can do over thirty days assuming that can do 2^{32} operations per second.
With $complexity_{i1} = 2^{2888}$ for all of the generated outputs we can say that they fulfill this condition. As for $complexity_{ii2}$ the results are the following[1]:
From the 50 generated samples the average $complexity_{ii2}$ is ≈ 0.0175181, the minimum was 0.00347231, and the maximum was 0.0455996352
These results are pretty low if we consider that the maximum value is 1. Finally we can use the equation for novelty:

$$\begin{aligned} novelty1(x) = 1 \text{ if } complexity_i > \alpha \\ \text{and } complexity_{ii} > \beta;\ 0 \text{ otherwise.} \end{aligned} \quad (5)$$

Relative to the average we can see that if we put a threshold $\beta = 0.03$ makes less products to be novel, compared to using the mean. Using the mean we end up with 18 novel products. The stats for how many actual novel products produced over the total 50 using $\beta = 0.03$ is 5.

Value Within Alison framework we will use the value measure relative to its aim that is to resemble of a work of Bach, which can then verified with a negative log likelihood on the probability of each score being generated given the training data.
Said results are stated over a generation of 100 pieces of length 32.

Sampling scheme	**Framewise NLL**
Ancestral Gibbs, $\rho = 0.00$ (NADE)	1.09 ± 0.06
Ancestral Gibbs, $\rho = 0.05$	1.08 ± 0.06
Ancestral Gibbs, $\rho = 0.10$	0.97 ± 0.05
Ancestral Gibbs, $\rho = 0.25$	0.80 ± 0.04
Ancestral Gibbs, $\rho = 0.50$	0.74 ± 0.04
Independent Gibbs	0.52 ± 0.01

Table 1: Mean ($\pm$ SEM) NLL under model of unconditioned samples generated from model by various schemes.

With these two evaluations of novelty and value where overall were a pretty low novelty and a fairly good value results on NLL we can conclude that the COCONET lacks of creative potential by generating on average not novel products relative to its complexity but that could be explained by its great quality in being similar to those of Bach which restricts the exploratory area in the conceptual space.

Evaluation of the process

Using the criteria described on (Ventura 2016) we could think of the system as a two section architecture. First we have the training section of the system. This section's goal

[1]The code repository for generation and evaluation can be found in `https://github.com/Sirivasv/CCEvaluation`

is not to generate products but to infer the underlying rules of music composition used by Bach. This section in particular could fit in ”Filtration” in the spectrum of candidate computationally creative processes. This is because of the presence of the fitness function while generating partial solutions in order to generalize the model based on the inspiring set. In the second section of the architecture, the sampling section, we can actually start generating the end products. In the spectrum it partially fits ”Inception” as it doesn’t use the inspiration set anymore, but the generative probability distributions inferred in the training section is the knowledge base used for the generation, because it implicitly describes the compositional structure used by Bach and wasn’t previously incorporated to the system. But considering the two sections as one system it nows fulfills the ”Inception” step in the spectrum as it uses the inspiration set to infer its own knowledge base at different stages.
Being too loose in the actual criteria to meet the ”Creation” step in the spectrum we could consider that the interaction with the user at harmonizing its melodies is a kind of perception of the external world. But it is rather an interface of communication for users with the system that an actual perception of the world for the system. The incorporation of some sort of translation capable peripherals of video, audio or symbolic music representations found in the external world could improve the fitness of the system in the ”Creation” step in the spectrum.

Development of an extension interface

On its original form the COCONET implementation is a python module included in the library Magenta, and to use it one can adjust a plethora of parameters in order to fine-tune its training stages and sampling procedures, but for the user interested only on its musical capabilities those can be ignored and just use the parameters which yielded the best results in the technical evaluation. The music composition workflow using COCONET is as follows:

1. Gather the trained COCONET or train a new one using the available dataset.
2. Gather a prime melody MIDI File if needed.
3. Run a python script providing the model, MIDI file (optional) and desired segment duration.
4. Locate and use the output of the system in a specified output directory.

For a non-technical musician this workflow represents a daunting task with a complicated learning curve to just use the system with the minimal adjustments. Now let’s take a look to the MuseScore interface in Figure 3, it will be a more familiar welcome to the task of music composition than a command line programming interface such as Python. Although easier it still will require some technical knowledge from the user as it is required to be installed and be configured to save the compositions in music specific file formats.

While having to learn how to use MuseScore it could be easier due the presence of fellow musicians who could help

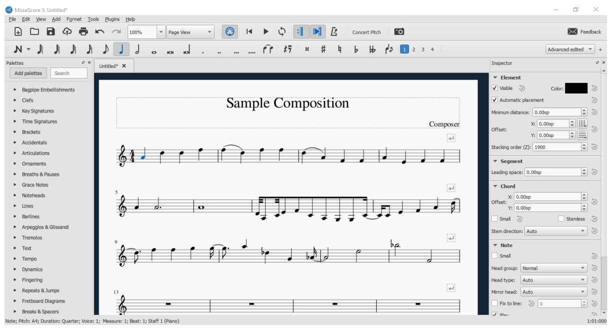


Figure 3: MuseScore Interface.

non-technical musicians to learn it rather than the actual developers of the tool. In the MuseScore Plug-in Environent a third-party plug-in can be easily installed by just downloading the required files in the plug-in directory of MuseScore. Having as an antecedent the work done in DeepBach (Hadjeres, Pachet, and Nielsen 2017), one can use and tweak its source code to develop new interfaces. By following the conventions in DeepBach the MuseScore plug-in of COCONET has this workflow:

1. Gather COCONET URL or initiate a local instance.
2. Initialize a new file in MuseScore with the desired 4 voices.
3. Write the melody segments to be harmonized.
4. Open the coconet plugin.
5. Select the melody segments to be harmonized.
6. Click the *Harmonize* button in the plugin interface.
7. After waiting the response 2 new files appear:
   - The melody file containing the sent melody.
   - The Harmonized melody.
8. Iteratively the user now can incorporate the response into his composition and repeat from the 3rd step until the completion of the musical score.

The plug-in interface alone can be seen in Figure 4, and the incorporation with MuseScore in Figure 5 which also displays an initial file without being harmonized and a selected melody segment. In Figure 6 it is now shown the sent segment along with an updated message and waiting animation from the plug-in. Finally in Figure 7 we can see the harmonization response.

As future work or improvements we can set the preferences for an user that maybe want to incorporate automatically the response into the current file or not to see the verbosity to display as separate files the sent segment and the response. Unfortunately this at the current state of the MuseScore Software is not possible, but by handling such task to the harmonization server could be of help. The server is another

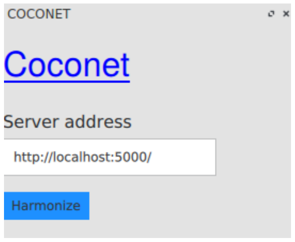


Figure 4: COCONET Plug-In Interface.

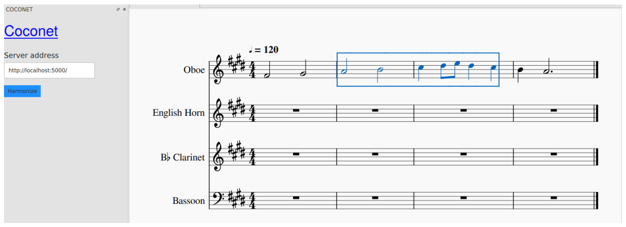


Figure 5: COCONET Plug-In Interface along MuseScore melody selection.

aspect as if a local instance is going to be used then it can imply some technical knowledge from the user but this can be solved a public url to receive melodies to be harmonized.

Conclusion

An extension of the artificial harmonizer COCONET was developed after a creativity evaluation of the original system. This extension suffered the lack of features in the plug-in development environment in MuseScore, nonetheless a prototype was able to perform the core task effectively in order to foster a more interactive interface between composers and AI powered creativity support tools which can be further improved to be considered partners in creative tasks. I believe that the intention was successfully fulfilled in providing an example for seeking a broader niche of users with all kinds of expertise in the domain, in this case from programmers to musicians with no technical experience. I hope this example be of inspiration for such seek of broader and welcoming niche of users in future works.

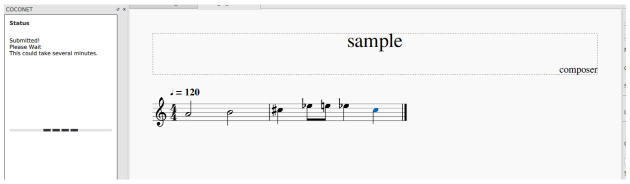
References

Allan, M., and Williams, C. 2005. Harmonising Chorales by Probabilistic Inference. In Saul, L. K.; Weiss, Y.; and Bottou, L., eds., *Advances in Neural Information Processing Systems 17*. MIT Press. 25–32.

Bazin, T., and Hadjeres, G. 2019. NONOTO: A Model-agnostic Web Interface for Interactive Music Composition by Inpainting. In *ICCC*.

Boden, M. A. 1991. *The creative mind: Myths & mechanisms*. The creative mind: Myths & mechanisms. New York, NY, US: Basic Books. Pages: xii, 303.

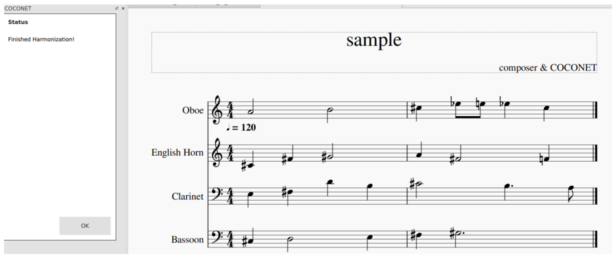
Cope, D. 1992. Computer Modeling of Musical Intelligence in EMI. *Computer Music Journal* 16(2):69–83. Publisher: The MIT Press.



Status

Submitted!
Please Wait
This could take several minutes.

Figure 6: COCONET Plug-In updated message and waiting animation along the sent melody segment.



Status

Finished Harmonization!

Figure 7: COCONET Harmonization.

Davis, N.; Hsiao, C.-P.; Popova, Y.; and Magerko, B. 2015. An Enactive Model of Creativity for Computational Collaboration and Co-creation. In Zagalo, N., and Branco, P., eds., *Creativity in the Digital Age*, Springer Series on Cultural Computing. London: Springer. 109–133.

Ebcioglu, K. 1990. An expert system for harmonizing chorales in the style of J.S. Bach. *The Journal of Logic Programming* 8(1):145–185.

Hadjeres, G.; Pachet, F.; and Nielsen, F. 2017. DeepBach: a Steerable Model for Bach Chorales Generation. *arXiv:1612.01010 [cs]*. arXiv: 1612.01010.

Hertzmann, A. 2018. Can Computers Create Art? *arXiv:1801.04486 [cs]*. arXiv: 1801.04486.

Hild, H.; Feulner, J.; and Menzel, W. 1992. HARMONET: A Neural Net for Harmonizing Chorales in the Style of J. S. Bach. In Moody, J. E.; Hanson, S. J.; and Lippmann, R. P., eds., *Advances in Neural Information Processing Systems 4*. Morgan-Kaufmann. 267–274.

Huang, C.-Z. A.; Cooijmans, T.; Roberts, A.; Courville, A.; and Eck, D. 2019a. Counterpoint by Convolution. *arXiv:1903.07227 [cs, eess, stat]*. arXiv: 1903.07227.

Huang, C.-Z. A.; Hawthorne, C.; Roberts, A.; Dinculescu, M.; Wexler, J.; Hong, L.; and Howcroft, J. 2019b. The Bach Doodle: Approachable music composition with machine learning at scale. *In ISMIR*.

Liang, F.; Gotham, M.; Johnson, M.; and Shotton, J. 2017. Automatic Stylistic Composition of Bach Chorales with Deep LSTM. In *ISMIR*.

Liu, J. S. 1994. The Collapsed Gibbs Sampler in Bayesian Computations with Applications to a Gene Regulation Problem. *Journal of the American Statistical Association* 89(427):958–966. Publisher: [American Statistical Association, Taylor & Francis, Ltd.].

Lubart, T. 2005. How can computers be partners in the creative process: Classification and commentary on the Special Issue. *International Journal of Human-Computer Studies* 63(4):365–369.

McIntyre, R. 1994. Bach in a box: the evolution of four part Baroque harmony using the genetic algorithm. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, 852–857 vol.2.

Meade, N.; Barreyre, N.; Lowe, S. C.; and Oore, S. 2019. Exploring conditioning for generative music systems with human-interpretable controls. In *ICCC*.

Pachet, F., and Roy, P. 2014. Non-conformant harmonization: the real book in the style of take 6. In *ICCC*.

Pease, A.; Winterstein, D.; and Colton, S. 2001. Evaluating Machine Creativity. In *ICCBR'01 Workshop on Creative Systems*, 9.

Pérez, R. P. y. 2018. The Computational Creativity Continuum. In *ICCC*.

Uria, B.; Côté, M.-A.; Gregor, K.; Murray, I.; and Larochelle, H. 2016. Neural Autoregressive Distribution Estimation. *arXiv:1605.02226 [cs]*. arXiv: 1605.02226.

Uria, B.; Murray, I.; and Larochelle, H. 2014. A Deep and Tractable Density Estimator. *arXiv:1310.1757 [cs, stat]*. arXiv: 1310.1757.

Ventura, D. 2016. Mere Generation: Essential Barometer or Dated Concept? In *ICCC*, 8.

Ventura, D. 2017. How to Build a CC System Paper type: System Description Paper. *In ICCC* 8.

Wiggins, G. A. 2006. A preliminary framework for description, analysis and comparison of creative systems. *Knowledge-Based Systems* 19(7):449–458.

Yao, L.; Ozair, S.; Cho, K.; and Bengio, Y. 2014. On the Equivalence Between Deep NADE and Generative Stochastic Networks. *arXiv:1409.0585 [cs, stat]*. arXiv: 1409.0585.